

# 1 AST

The AST is the internal representation of a proof program.

---

```
module AST where
data AST = Scope [AST] -- DeclList
| ID String AST -- name ArgumentList
| ArgumentList [AST] -- [Annotation]
| TypeOf AST -- Value
| Annotation String AST -- name Type
| Let AST AST AST -- ID Type Body
| Arrow AST AST -- Annotation Type
| Function AST AST -- ID Body
| Application AST AST -- Function Value
| Exists AST AST -- Annotation Body
| IntroExists AST AST -- Type Value [will this need another
  argument?]
| ElimExists AST AST -- Exists Body [how does this work again?
  does it need another argument too?]
| And AST AST -- Type Type
| IntroAnd AST AST -- Left Right
| ElimAndLeft AST AST -- And Body
| ElimAndRight AST AST -- And Body
| Or AST AST -- Type Type
| IntroOrLeft AST AST -- Or Value
| IntroOrRight AST AST -- Or Value
| ElimOr AST AST AST -- Or LeftBody RightBody
| Contradiction
| ElimContradiction AST AST -- Contradiction Body [does this
  have a body? contradiction usually means done]
-- [will this need equality type and reflexivity?]
-- value nodes
| VNatural Int -- Value
| VFloat Float -- Value [is this needed? or just define as a
  pair or in STL]
| VChar Char -- Value [is this needed? or just define as an
  int or in STL]
| VBoolean Bool -- True/False
| VCons AST AST -- Head Tail
| VEmpty -- empty list
| VSymbol String -- For
| VNull -- the empty value
| VUndefined -- the non-existent value
-- induction [do these need that 4th param like last time?]
-- this needs a way to write induction in the code!!
| IndNatural AST AST AST -- Int BodyS BodyZ
| IndBoolean AST AST AST -- Bool BodyT BodyF
| IndList AST AST AST -- List BodyL BodyE [is this correct?]
```

```
| Insert  
| BuiltIn String -- name  
deriving (Show, Eq)
```

---